IN THE CLAIMS

Please amend the claims as follows.

1. (Currently amended) A computer implemented method comprising:

beginning initialization of a first thread from a second context, wherein the first thread is ~~capable of being executed~~ executable on a first context and the second context;

suspending the initialization of the first thread at a position within the second context, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread;

creating a second thread based on the position in the second context; and

completing the initialization of the first thread continuing from the position in the second context.

2. (Original) The method of claim 1, wherein the beginning initialization of the first thread includes allocating per-thread context resources.

3. (Canceled)

4. (Original) The method of claim 1, wherein the second context is the platform-independent code to be executed on a host platform.

5. (Original) The method of claim 1, wherein the second context is a host platform that supports multiple instruction set architectures (ISA).

6. (Original) The method of claim 1, wherein completing the initialization of the first thread includes executing an application programming interface that makes an operating system request to create the first thread in the second context.

7. (Original) The method of claim 1, wherein the first context is an IA-32 multithreaded program and the second context is an IA-64 multithreaded program.

8. (Original) The method of claim 1, wherein the first context is an IA-32 platform and the second context is an IA-64 platform.

9. (Currently amended) A computer implemented method comprising:

beginning initialization of a foreign thread from a host platform, wherein the foreign thread is to be executed on a foreign platform;

suspending the initialization of the foreign thread at a position within the host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread;

recording the position of the suspension;

creating a host thread from a host platform based on the recorded position; and

completing the initialization of the foreign thread continuing from the recorded position in the host platform.

10. (Original) The method of claim 9, wherein the beginning initialization of the foreign thread includes allocating per-thread context resources.

11. (Canceled)

12. (Original) The method of claim 9, wherein the host platform supports platform-independent code.

13. (Original) The method of claim 9, wherein the host platform supports multiple instruction set architectures (ISA).

14. (Original) The method of claim 9, wherein the foreign platform is a IS-32 platform and the host platform is a IS-64 platform.

15. (Currently amended) A computer system for managing thread ~~resource~~ resources comprising:

a first multithreaded programming environment, executed by a processor;

a second multithreaded programming environment, executed by the processor;

a multithreaded program, executed by the processor, including a first thread and a second thread;

a host platform; and

a dynamic binary translator to ~~manage and support~~ translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment, wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected.

16. (Currently amended) The system of claim 15 further ~~comprises~~ comprising a first component to provide a communication interface between the dynamic binary translator and the multithread programming environment.

17. (Currently amended) The system of claim 15 further ~~comprises~~ comprising a first thread library and a second thread library.

18. (Canceled)

19. (Currently amended) A computer system for managing thread resource comprising:

a random accessed memory;

a first processor ~~capable of executing~~ configured to execute multithreaded programs stored in the random accessed memory;

a multithreaded program to be executed on a second processor; and

a program to transparently initialize and create a thread included in the multithreaded program in an environment supported by the first processor to be executed on the second processor, wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread.

20. (Original) The system of claim 19, wherein the program further allocates per-thread context resources.

21. (Original) The system of claim 20, wherein the program initializes and creates the thread transparently by associating the allocated per-thread context resource between the environment supported by the first processor.

22. (Canceled)

23. (Canceled)

24. (Canceled)

25. (Canceled)